(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
10 November 2005 (10.11.2005)

PCT

(10) International Publication Number
WO 2005/106646 A1

(51) International Patent Classification⁷: G06F 9/32, 9/302, 9/30

(21) International Application Number: PCT/US2002/020774

(22) International Filing Date: 21 November 2001 (21.11.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/748,165 27 December 2000 (27.12.2000) US

(71) Applicant (*for all designated States except US*): **INTEL CORPORATION** [US/US]; 2200 Mission College Boulevard, Santa Clara, CA 95052 (US).

(72) Inventor; and

(75) Inventor/Applicant (*for US only*): **PAVER, Nigel, C.** [GB/US]; 10200 Damlgreen Avenue, Austin, TX 78739 (US).

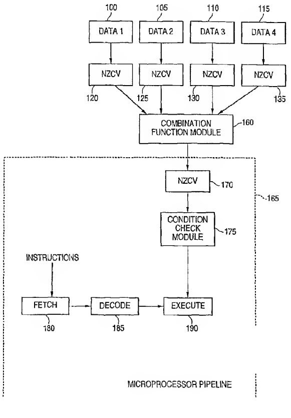
(74) Agents: **SKWIERAWSKI, Paul, J.** et al.; Antonelli Terry Stout Kraus, 1300 North Seventeenth Street, Suite 1800, Arlington, VA 22209 (US).

(81) Designated States (*national*): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZM, ZW.

(84) Designated States (*regional*): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent

[Continued on next page]

(54) Title: METHOD AND COMPUTER PROGRAM FOR SINGLE INSTRUCTION MULTIPLE DATA MANAGEMENT



(57) Abstract: A method and computer program for extracting and combining arithmetic flags utilized in the processing multiple data items in a single instruction multiple data (SIMD) capable processor. In a SIMD processor several pieces of data may be manipulated by the same instruction at any given moment. However, the results for the execution of this instruction vary according to the data being manipulated. The method and computer program allows a simple mechanism in which these arithmetic flags maybe extracted and combined so as to maximize processor efficiency while saving space, reducing power requirements and heat generated by the processor.

WO 2005/106646 A1

(BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

METHOD AND COMPUTER PROGRAM FOR SINGLE INSTRUCTION MULTIPLE DATA MANAGEMENT

FIELD

The invention relates to a method and computer program for single instruction multiple data (SIMD) management. More particularly, the present invention manages the arithmetic flags associated with individual data items so that a processor with SIMD capability may logically combine these arithmetic flags so that simultaneous processing of multiple data items may be done at the same time in a simple and efficient manner.

BACKGROUND

In the rapid development of computers many advancements have been seen in the areas of processor speed, throughput, communications, and fault tolerance. Initially computer systems were standalone devices in which a processor, memory and peripheral devices all communicated through a single bus. Later, in order to improve performance, several processors were interconnected to memory and peripherals using one or more buses. In addition, separate computer systems were linked together through different communications mechanisms such as, shared memory, serial and parallel ports, local area networks (LAN) and wide area networks (WAN). Further, in order to improve processor instruction processing, pipelining was developed to enable a processor to execute an instruction in stages and a single processor could execute different instructions at different stages of execution simultaneously.

A further development created in order to enhance processor performance is the use of a technique known as single instruction multiple data (SIMD). SIMD is a technique where several different pieces of data may be simultaneously accessed and arithmetically manipulated by a processor. This ability to manipulate several pieces of data at the same time greatly enhances the performance of the processor. However, even though the same arithmetic operation may be performed, the results and status for each piece of data may be different. For example, the data may be negative, zero, have a carry out or overflow condition resulting. Since a SIMD processor may manipulate as many as eight pieces, or more, of data simultaneously, the processor is required to maintain at least eight sets of these condition flags. Further, in order to receive the benefit of SIMD processing it is necessary to logically combine these condition or arithmetic flags so that the appropriate operation may occur under the appropriate conditions. Since it may be necessary to manipulate eight pieces, or more, of data under many different combinations of possible outcomes, the logic that must be built into a processor and microprocessor design can be very cumbersome. Valuable space on the microprocessor must be dedicated to this processing and the speed, size, power required, and heat generated by the processor may be seriously effected.

Therefore, what is needed is a method and computer program which will combine the arithmetic or condition flags in a simple manner so that the appropriate operation will be performed under the appropriate conditions. Further, this method and computer program should allow for the testing of all arithmetic functions and condition flags at once in a simple manner. In addition, this method and computer program should be able to simply extract individual arithmetic flags for individual data items when necessary.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and a better understanding of the present invention will become apparent from the following detailed description of exemplary embodiments and the claims when read in connection with the accompanying drawings, all forming a part of the disclosure of this invention. While the foregoing and following written and illustrated disclosure focuses on disclosing example embodiments of the invention, it should be clearly understood that the same is by way of illustration and example only and the invention is not limited thereto. The spirit and scope of the present invention are limited only by the terms of the appended claims.

The following represents brief descriptions of the drawings, wherein:

FIG. 1A is an example embodiment of the arithmetic flags in an SIMD word for eight data items stored in a processor status register (PSR) used in an example embodiment of the present invention;

FIG. 1B is an example embodiment of the arithmetic flags in an SIMD word for four data items stored in a PSR used in example embodiment of the present invention;

FIG. 1C is an example embodiment of the arithmetic flags in an SIMD word for two data items stored in a PSR used in an example embodiment of the present invention;

FIG. 1D is an example embodiment of the arithmetic flags in an SIMD word for one data item stored in a PSR used in an example embodiment of the present invention;

FIG. 2 is a systems diagram of an example embodiment of the present invention;

FIG. 3 is an example flowchart of a general embodiment of the present invention;

FIG. 4 is a flowchart of an AND function used in an example embodiment of the present invention;

FIG. 5 is a flowchart of an OR function used in an example embodiment of the present invention; and

FIG. 6 is a flowchart of an EXTRACT function used in an example embodiment of the present invention.

DETAILED DESCRIPTION

Before beginning a detailed description of the subject invention, mention of the following is in order. When appropriate, like reference numerals and characters may be used to designate identical, corresponding or similar components in differing figure drawings. Further, in the detailed description to follow, exemplary sizes/models/values/ranges may be given, although the present invention is not limited to the same. As a final note, well-known components of computer networks may not be shown within the FIGs. for simplicity of illustration and discussion, and so as not to obscure the invention.

FIG. 1A through 1D are representative examples of SIMD words utilized to indicate the arithmetic flags associated with data items being manipulated by a processor having SIMD capability in the example embodiments of the present invention. FIG. 1A represents an SIMD word having eight sets of SIMD flags contained therein labeled 120, 125, 130, 135, 140, 145, 150 and 155. Each SIMD set (120, 125, 130, 135, 140, 145, 150 and 155) has four variables associated with it designated N, Z, C, and V. N represents a data item which has a negative value. Z represents a data item which has a value of zero. C represents a carry out condition in a data item which would occur in the case of an overflow for a byte or word having a sign bit. V represents an overflow condition having occurred for an associated data item. It should be noted that N, Z, C, and V are only examples of arithmetic flags. As would be appreciated by one of ordinary skill in the art many more such flags or conditions may be created for results generated by arithmetic functions. Therefore, the flags indicated in FIGs. 1A through 1D are provided as examples only and it is not intended that the present invention be limited the use of these flags or conditions only.

Referring to FIG. 1A, eight sets of arithmetic flags (120, 125, 130, 135, 140, 145, 150 and 155) are shown in which each set of flags is associated with an individual data item. Therefore, the first set of flags composed of N, Z, C, and V is associated with the first data item 120 while the second 125, third 130, and fourth 135 through eighth 155 are associated with the first, second, third, and fourth through eighth data items further illustrated in FIG. 2 and discussed ahead. It should be noted that this particular SIMD word contains 32 bits. However, the present invention is not restricted to the use of a 32-bit SIMD word . It is possible for a 64-bit SIMD word to be utilized in which the embodiments of the present invention may utilize this 64-bit SIMD word to operate.

Referring to FIG. 1B, it should be noted that the SIMD word illustrated is similar to that shown in FIG. 1A, however, only four sets of arithmetic flags (120, 125, 130 and 135) are set. As with FIG. 1A, the same N, Z, C, and V designation is used with the exception that each byte has the least significant bits occupied by the value zero.

Referring to FIG. 1C, this figure is similar to FIG. 1A and FIG. 1B with the exception that only two sets of arithmetic flags (120 and 125) are represented. Therefore, each of the least significant bits not used in each half word are filled with value zero.

Referring to FIG. 1D, this figure is similar to FIG. 1A, 1B, and 1C with the exception that only one set of arithmetic flags (120) are represented. Therefore, each of the least significant bits not used in each word are filled with value zero.

FIG. 2 is a systems diagram of an example embodiment of the present invention. As illustrated in FIG. 1B, arithmetic flags 120, 125, 130 and 135 are shown in FIG. 2. However, in addition arithmetic flags 120, 125, 130 and 135 are each associated with data items 100, 105, 110 and 115 respectively. As previously discussed, in order for a SIMD capable processor, such as processor 165, to effectively be able to manipulate multiple pieces of data (100 B 115) it is necessary to logically combine the results of mathematical operations shown in arithmetic flags 100, 125,130 and 135. This is accomplished by the combination function module 160 utilizing the methods and operations illustrated and further discussed in reference to FIGs. 3 B 6. The results of the combination function performed by the combination function module 160 is a combined arithmetic flag variable 170. Thereafter, a condition check module 175 is utilized to determine the next operation to perform based upon the combined arithmetic flag variable 170. These operations will be discussed further detail ahead.

Still referring to FIG. 2, as discussed earlier, pipelining is a common form of computer architecture. In processor 165 at least three stages of pipelining are shown. The first stage of pipelining is the fetch 180 operation in which instructions are retrieved from memory (not shown) for execution. The second stage of pipelining is a decode operation 185 in which the instruction is decoded by the processor. Finally, the last stage of this example processor pipeline is the execute 190 stage in which the instruction is executed based upon input from the condition check module 175. As would be appreciated by one of ordinary skill in the art, the example processor pipeline shown in FIG. 2 is merely an example. Many more stages of pipelining are possible.

Before proceeding into a detailed discussion of the logic used by the present invention it should be mentioned that the flowcharts shown in FIGs. 3 through 6 or contain software, firmware, hardware, processes or operations that correspond, for example, to code, sections of code, instructions, commands, objects, hardware or the like, of a computer program that is embodied, for example, on a storage medium such as floppy disk, CD-Rom (Compact Disc read-only Memory), EP-Rom (Erasable Programmable read-only Memory), RAM (Random Access Memory), hard disk, etc. Further, the computer program can be written in any language such as, but not limited to, for example C ++. Further, the logic shown in figs 3 B 6 are executed by the modules and processor 165 shown in FIG. 2.

FIG. 3 is an of an example flowchart of a general embodiment of the present invention. Logic utilized in the flowchart illustrated in FIG. 3 maybe used to combine, group, or extract the arithmetic flags illustrated in FIGs. 1A through 1B. The functions that may be executed by the condition check module 175 would include, but not be limited to, the following functions.

1. If any field has overflowed;
2. If any field has not overflowed;
3. If any field is positive (or zero);
4. If any field is negative;

5. If any field is zero;
6. If any field is not zero;
7. If any field has a carry out;
8. If any field does not have a carry out;
9. If all fields have overflowed;
10. If all fields have not overflowed;
11. If any field are positive (or zero);
12. If all fields are negative;
13. If all fields are zero;
14. If all fields are not zero;
15. If all fields have a carry out; and
16. If all fields do not have a carry out.

As would be appreciated by one order skill of the art the foregoing functions may be increased to include any mathematical functions including less than, greater than, less than or equal to, and greater than or equal to. Additional, mathematical operators and functions may be used in conjunction with the present invention.

Still referring to FIG. 3, processing begins in operation 200 and immediately proceeds operation 210. In operation 210, a field size is determined on which to base the extraction or combination function. The field size may be, but not limited to, a nibble, byte, half word, word, or double word in size. The extraction and/or combination function may include any of the foregoing 16 items discussed or any other function which may describe or combine the status or result of a mathematical operation performed by a computer or processor. Thereafter, processing proceeds operation 220 where it is determined if an extraction process is being performed. If an extraction process is being performed processing then proceeds operation 230. In operation 230, the flags, illustrated in FIGs. 1A through 1D, are extracted based upon the field size determined in operation 210 and the specific data item desired. Thereafter, processing proceeds operation 270 where the extracted information is stored in the destination register. Once stored processing proceeds to operation 280 where processing terminates. In an example embodiment shown in FIG. 6, the extraction process is further detailed as discussed ahead.

If in operation 220 it is determined that an extraction process is not desired, then processing proceeds operation 240. In operation 240 it is determined whether a combination process executed by the condition check module 175 for the arithmetic flags illustrated in FIGs. 1A through 1D is desired. If a combination process is not desired then processing proceeds operation 280 where again processing terminates. However, if a combination process executed by the condition check module 175 is desired for the flags associated with several data items shown in FIGs. 1A through 1D, then processing proceeds operation 250. In operation 250, the flags for each data item in the SIMD PSR register are extracted based on the field size determined in operation 210. Processing then proceeds to operation 260 where the extracted flags for each data item are combined based upon the function desired. Specific examples of combination functions for an AND operation and an OR operation are further detailed in the discussion of FIG. 4 and FIG. 5, respectively. Thereafter, processing proceeds to operation 270 where the results of the combined flags are stored in the destination register for access by the processor. Processing then terminates in operation 280.

FIG. 4 is an of a flowchart of an AND function used in an example embodiment of the present invention and may be executed by the condition check module 175. Processing for this AND operation begins in operation 300 and immediately proceeds operation 310. In operation 310 it is determined whether the data field size is four bits (one nibble) in length. If the data field size is four bits in length then processing proceeds to operation 320. In operation 320, bits 31 through 28 of the destination register are set equal to bits 31 through 28 anded with bits 27 through 24 anded with bits 23 through 20 anded with bits 19 through 16 anded with bits 15 through 12 anded with bits 11 through 8 anded with the 7 through 4 and 3 through 0 of the SIMD PSR register. Thereafter, processing proceeds to operation 320 where the remaining bits 27 through 0 of the destination register are set to zero. Processing then proceeds to operation 395 where processing terminates.

Still referring to FIG. 4, if in operation 310 it is determined that a four bits data field is not specified then processing proceeds to operation 340. In operation 340, it is determined whether an 8 bit (byte) data field is specified. If an 8 bit data field is specified in the SIMD data word, shown in FIG. 1B, then processing proceeds to operation 350. In operation 350, bits 31 through 24 of the destination register are set equal to bits 31 through 24 anded with bits 23 through 16 anded with bits 15 through 8 and bits 7 through 0 of the SIMD PSR register. Thereafter, processing proceeds to operation 360 where bits 23 through 0 of the destination register are set to zero. Processing then terminates in operation 395.

Still referring to FIG. 4, if in operation 340 it is determined that an 8 bit data field is not specified, then processing proceeds operation 370. In operation 370 it is determined whether a 16-bit (half word) data field is specified. If a 16-bit data field is specified, as shown in FIG. 1C, then processing proceeds to operation 380. In operation 380, bits 31 through 16 of the destination register are set equal to bits 31 through 16 anded with bits 15 through 0 of the SIMD PSR register. Thereafter, processing proceeds to operation 390 where bits 15 through 0 of the destination register are set to zero. Then, in operation 395, processing is terminated.

FIG. 5 is an of a flowchart of an OR function used in an example embodiment of the present invention and may be executed by the condition check module 175. Processing for this OR operation begins in operation 400 and immediately proceeds operation 410. In operation 410 it is determined whether the data field size is four bits (one nibble) in length. If the data field size is four bits in length then processing proceeds to operation 420. In operation 420, bits 31 through 28 of the destination register are set equal to bits 31 through 28 ORD with bits 27 through 24 ORD with bits 23 through 20 ORD with bits 19 through 16 ORD with bits 15 through 12 ORD with bits 11 through 8 ORD with the 7 through 4 ORD with 3 through 0 of the SIMD PSR register. Thereafter, processing proceeds to operation 420 where the remaining bits 27 through 0 of the destination register are set to zero. Processing then proceeds to operation 495 where processing terminates.

Still referring to FIG. 5, if in operation 410 it is determined that a four bits data field is not specified, then processing proceeds to operation 440. In operation 440, it is determined whether an 8 bit (byte) data field is specified. If an 8 bit data field is specified in the SIMD data word shown in FIG. 1B, then processing proceeds to operation 450. In operation 450, bits 31 through 24 of the destination register are set equal to bits 31 through 24 ORD with bits 23 through 16 ORD with bits 15 through 8 ORD with bits 7 through 0 of the SIMD PSR register. Thereafter, processing proceeds to operation 460 where bits 23 through 0 of the destination register are set to zero. Processing then terminates in operation 495.

Still referring to FIG. 5, if in operation 440 it is determined that an 8 bit data field is not specified, then processing proceeds operation 470. In operation 470 it is determined whether a

16-bit (half word) data field is specified. If a 16-bit data field is specified, as shown in FIG. 1C, then processing proceeds to operation 480. In operation 480, bits 31 through 16 of the destination register are set equal to bits 31 through 16 ORD with bits 15 through 0 of the SIMD PSR register. Thereafter, processing proceeds to operation 490 where bits 15 through 0 of the destination register are set to zero. Then in operation 495 processing is terminated.

FIG. 6 is a flowchart of an EXTRACT function used in an example embodiment of the present invention and may be executed by the condition check module 175. The extract function begins execution in operation 500 and immediately proceeds to operation 510. In operation 510, it is determined whether the data field illustrated in FIG. 1A for the SIMD word is four bits (one nibble) in length. If the data field is determined to be four bits in length, in operation 510, then processing proceeds operation 520. In operation 520, bits 31 through 28 of the destination register are set equal to nibble 2 through 0 of the SIMD PSR register. Thereafter, processing proceeds to operation 570 where processing terminates.

However, if in operation 510 it is determined the data field is not equal to four bits in length then processing proceeds to operation 530. In operation 530, it is determined whether the data field is eight bits (one byte) in length. If the data field in the SIMD word is eight bits in length, as shown in FIG. 1B, then processing proceeds to operation 540. In operation 540, bits 31 through 24 of the destination register are set equal to bytes 1 through 0 of the SIMD PSR register. Again, processing then proceeds to operation 570 where processing terminates.

Still referring to FIG. 6, if in operation 530 it is determined that the data field in the at SIMD word is not one byte in length, then processing proceeds to operation 550. In operation 550, it is determined whether the data field length in the SIMD word is 16 bits (half word) in length. If the data field in the SIMD word is 16 bits in length, then processing proceeds to operation 560. In operation 560, bits 31 through 16 of the destination register are set equal to half word 0 in the SIMD PSR register. Thereafter, processing proceeds to operation 570 where processing terminates. Further, if it is determined in operation 550 that the data field length of the at SIMD word is not 16 bits, then processing proceeds to operation 570 where processing terminates.

The benefit resulting from the present invention is that a simple, reliable, fast method and computer program is provided that will enable a SIMD capable processor of extracting and/or combining arithmetic flags associated with multiple data items that have been the subject of mathematical operations. This method and computer program is of such in nature that complex logic is not required thus saving space, power requirements and heat generated by a processor. Further, this method and computer program allows a SIMD capable processor of operating at peak efficiency due to the simplicity of the logic required.

While we have shown and described only a few examples herein, it is understood that numerous changes and modifications as known to those skilled in the art could be made to the example embodiment of the present invention. Therefore, we do not wish to be limited to the details shown and described herein, but intend to cover all such changes and modifications as are encompassed by the scope of the appended claims.

CLAIMS

I Claim:

1. A device for combining a plurality of arithmetic flags, comprising:

a combination function module that examines a plurality of arithmetic flags, determines field size of the plurality of arithmetic flags and based on the determination of the field size will combine the plurality of arithmetic flags into a single combined arithmetic flag variable, wherein the plurality of arithmetic flags represent the status of a plurality of data items after a mathematical operation is performed by the processor on the plurality of data items.

2. The device recited in claim 1, further comprising:

a condition check module that determines the status of the combined arithmetic flag variable and causes the processor to execute an appropriate operation based on the status.

3. The device recited in claim 1, wherein the field size is based either a nibble, byte, half word, or word in length.

4. The device recited in claim 3, wherein the plurality of arithmetic flags further comprise:

a negative data value, a zero data value, a carry out occurrence in a data value, or an overflow condition in a data item in the plurality of data items.

5. The device recited in claim 4, the combination function module performs either an AND or an OR operation.

6. The device recited in claim 2, wherein the status determined by the condition further comprises:

any data item has overflowed;

any data item has not overflowed;

any data item is positive or zero;

any data item is negative;

any data item is zero;

any data item is not zero;

any data item has a carry out;

any data item does not have a carry out;

all data items have overflowed;

all data items have not overflowed;

all data items are positive or zero;

all data items are negative;

all data items are zero;

all data items are not zero;

all data items have a carry out; and

all data items do not have a carry out.

7. A method of combining a plurality of arithmetic flags for presentation to a processor, comprising:

determining a field size of the plurality of arithmetic flags on which to base a combination process, wherein the plurality of arithmetic flags represent the status of a plurality of data items after a mathematical operation is performed by the processor on the plurality of data items;

extracting the plurality of arithmetic flags based on the field size;

combining the plurality of arithmetic flags based on a function selected when a combination process is selected; and

storing a result of the combining of the plurality of arithmetic flags in a destination register for access by the processor.

8. The method recited in claim 7, wherein the field size is based either a nibble, byte, half word, or word in length.

9. The method recited in claim 8, wherein the plurality of arithmetic flags further comprise:

a negative data value, a zero data value, a carry out occurrence in a data value, or an overflow condition in a data item in the plurality of data items.

10. The method recited in claim 9, wherein the function further comprises: an AND or OR operation.

11. The method recited in claim 10, wherein the function may be used to determine the status of the plurality of data items, said status comprising:

any data item has overflowed;

any data item has not overflowed;

any data item is positive or zero;

any data item is negative;

any data item is zero;

any data item is not zero;

any data item has a carry out;

any data item does not have a carry out;

all data items have overflowed;

all data items have not overflowed;

all data items are positive or zero;

all data items are negative;

all data items are zero;

all data items are not zero;

all data items have a carry out; and

all data items do not have a carry out.

12. An apparatus comprising a data storage medium for storing instructions when executed by a processor results in, comprising:

determining a field size of the plurality of arithmetic flags on which to base a combination process, wherein the plurality of arithmetic flags represent the status of a plurality of data items after a mathematical operation is performed by the processor on the plurality of data items;

extracting the plurality of arithmetic flags based on the field size;

combining the plurality of arithmetic flags based on a function selected when a combination process is selected; and

storing a result of the combining of the plurality of arithmetic flags in a destination register for access by the processor.

13. The apparatus recited in claim 12, wherein the field size is based either a nibble, byte, half word, or word in length.

14. The apparatus recited in claim 13, wherein the plurality of arithmetic flags further comprise:

a negative data value, a zero data value, a carry out occurrence in a data value, or an overflow condition in a data item in the plurality of data items.

15. The apparatus recited in claim 14, wherein the function further comprises an AND or OR operation.

16. The apparatus recited in claim 15, wherein the function may be used to determine the status of the plurality of data items, said status comprising:

any data item has overflowed;

any data item has not overflowed;

any data item is positive or zero;

any data item is negative;

any data item is zero;

any data item is not zero;

any data item has a carry out;

any data item does not have a carry out;

all data items have overflowed;

all data items have not overflowed;

all data items are positive or zero;

all data items are negative;

all data items are zero;

all data items are not zero;

all data items have a carry out; and

all data items do not have a carry out.

17. A method of extracting a plurality of arithmetic flags for presentation to a processor, comprising:

determining a field size of the plurality of arithmetic flags on which to base a combination process, wherein the plurality of arithmetic flags represent the status of a plurality of data items after a mathematical operation is performed by the processor on the plurality of data items;

extracting the plurality of arithmetic flags based on the field size; and

storing a result of the extracting of the plurality of arithmetic flags in a destination register for access by the processor.

18. The method recited in claim 17, wherein the field size is based either a nibble, byte, or half word in length.

19. The method recited in claim 18, wherein the plurality of arithmetic flags further comprise:

a negative data value, a zero data value, a carry out occurrence in a data value, or an overflow condition in a data item in the plurality of data items.

20. A method of extracting a plurality of arithmetic flags for presentation to a processor, comprising:

determining a field size of the plurality of arithmetic flags on which to base a combination process, wherein the plurality of arithmetic flags represent the status of a plurality of data items after a mathematical operation is performed by the processor on the plurality of data items;

extracting the plurality of arithmetic flags based on the field size; and

storing a result of the extracting of the plurality of arithmetic flags in a destination register for access by the processor.

21. The method recited in claim 20, wherein the field size is based either a nibble, byte, or half word in length.

22. The method recited in claim 21, wherein the plurality of arithmetic flags further comprise:

a negative data value, a zero data value, a carry out occurrence in a data value, or an overflow condition in a data item in the plurality of data items.

FIG. 1A

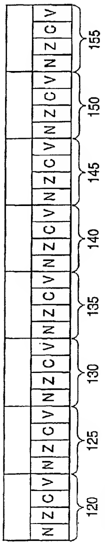


FIG. 1B

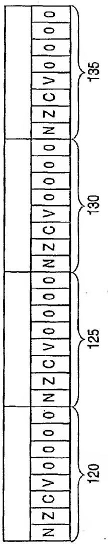


FIG. 1C

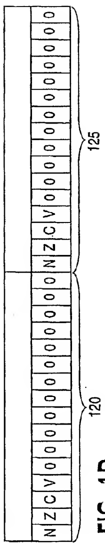


FIG. 1D

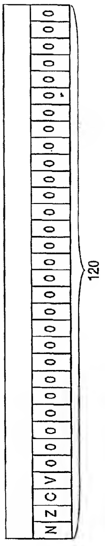


FIG. 2

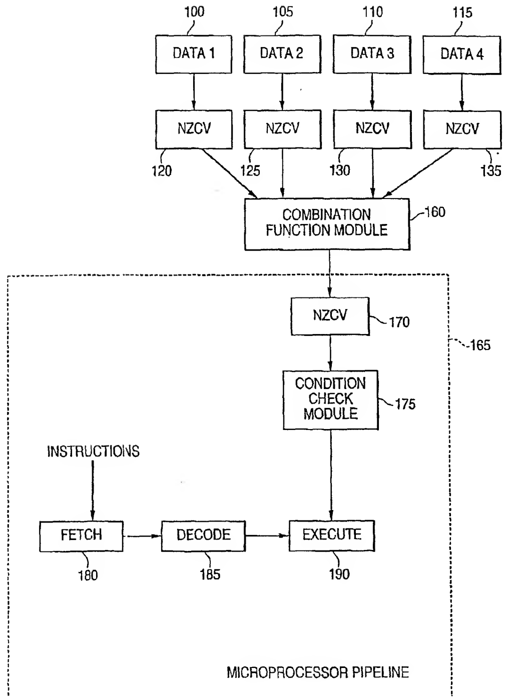

100
105
110
115
DATA 1
DATA 2
DATA 3
DATA 4
NZCV
NZCV
NZCV
NZCV
120
125
130
135
COMBINATION FUNCTION MODULE
160
NZCV
170
165
CONDITION CHECK MODULE
175
INSTRUCTIONS
FETCH
DECODE
EXECUTE
180
185
190
MICROPROCESSOR PIPELINE

FIG. 3

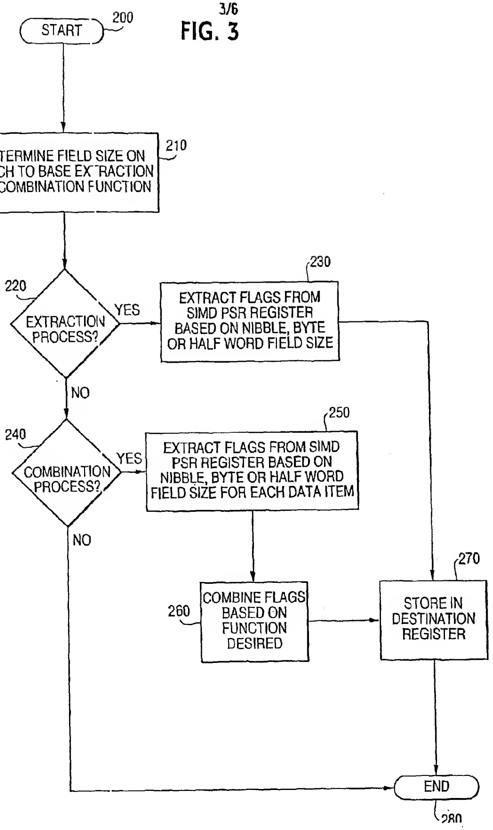

START
200
DETERMINE FIELD SIZE ON WHICH TO BASE EXTRACTION OR COMBINATION FUNCTION
210
220
EXTRACTION PROCESS?
YES
NO
EXTRACT FLAGS FROM SIMD PSR REGISTER BASED ON NIBBLE, BYTE OR HALF WORD FIELD SIZE
230
240
COMBINATION PROCESS?
YES
NO
EXTRACT FLAGS FROM SIMD PSR REGISTER BASED ON NIBBLE, BYTE OR HALF WORD FIELD SIZE FOR EACH DATA ITEM
250
COMBINE FLAGS BASED ON FUNCTION DESIRED
260
STORE IN DESTINATION REGISTER
270
END
280

FIG. 4

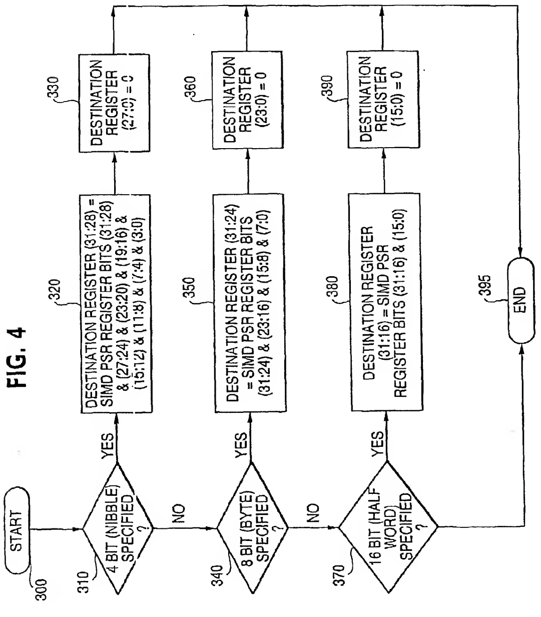
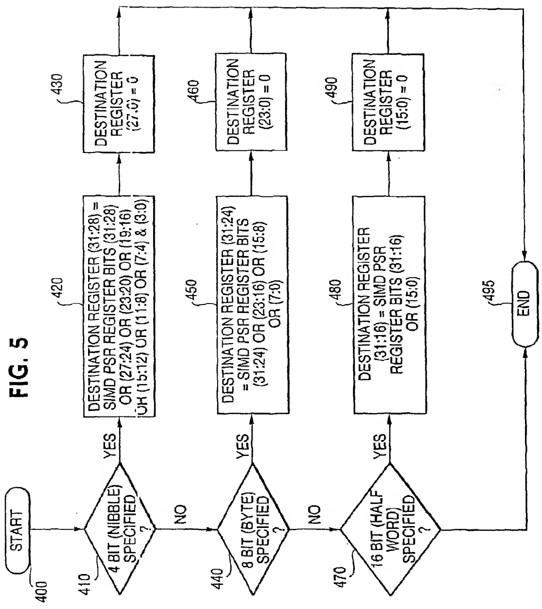

START
300
4 BIT (NIBBLE) SPECIFIED ?
310
YES
NO
DESTINATION REGISTER (31:28) = SIMD PSR REGISTER BITS (31:28) & (27:24) & (23:20) & (19:16) & (15:12) & (11:8) & (7:4) & (3:0)
320
DESTINATION REGISTER (27:0) = 0
330
8 BIT (BYTE) SPECIFIED ?
340
YES
NO
DESTINATION REGISTER (31:24) = SIMD PSR REGISTER BITS (31:24) & (23:16) & (15:8) & (7:0)
350
DESTINATION REGISTER (23:0) = 0
360
16 BIT (HALF WORD) SPECIFIED ?
370
YES
DESTINATION REGISTER (31:16) = SIMD PSR REGISTER BITS (31:16) & (15:0)
380
DESTINATION REGISTER (15:0) = 0
390
END
395

FIG. 5


START
400
4 BIT (NIBBLE) SPECIFIED ?
410
YES
NO
DESTINATION REGISTER (31:28) = SIMD PSR REGISTER BITS (31:28) OR (27:24) OR (23:20) OR (19:16) OR (15:12) OR (11:8) OR (7:4) & (3:0)
420
DESTINATION REGISTER (27:0) = 0
430
8 BIT (BYTE) SPECIFIED ?
440
YES
NO
DESTINATION REGISTER (31:24) = SIMD PSR REGISTER BITS (31:24) OR (23:16) OR (15:8) OR (7:0)
450
DESTINATION REGISTER (23:0) = 0
460
16 BIT (HALF WORD) SPECIFIED ?
470
YES
DESTINATION REGISTER (31:16) = SIMD PSR REGISTER BITS (31:16) OR (15:0)
480
DESTINATION REGISTER (15:0) = 0
490
END
495

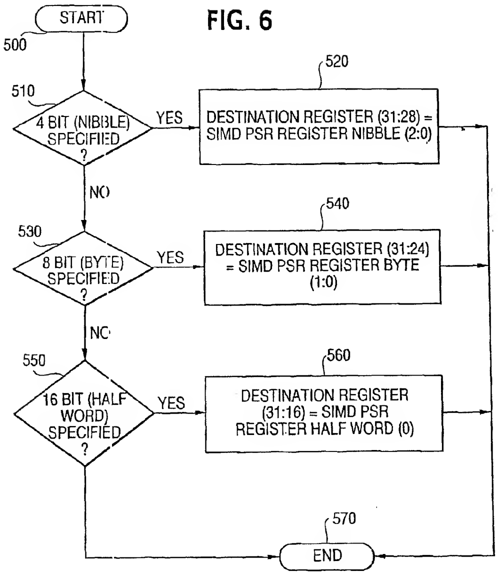

START
500
510
4 BIT (NIBBLE) SPECIFIED ?
YES
NO
520
DESTINATION REGISTER (31:28) = SIMD PSR REGISTER NIBBLE (2:0)
530
8 BIT (BYTE) SPECIFIED ?
YES
NO
540
DESTINATION REGISTER (31:24) = SIMD PSR REGISTER BYTE (1:0)
550
16 BIT (HALF WORD) SPECIFIED ?
YES
560
DESTINATION REGISTER (31:16) = SIMD PSR REGISTER HALF WORD (0)
570
END

FIG. 6

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 02/20774

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F9/32 G06F9/302 G06F9/30

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal, INSPEC, IBM-TDB

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 6 026 484 A (GOLSTON ET AL) 15 February 2000 (2000-02-15)	1-5, 7-10, 12-15, 17-22
A	column 5, line 36 - line 55 column 18, line 35 - column 23, last line column 26, line 28 - line 42 -----	6,11,16
A	US 5 815 723 A (WILKINSON ET AL) 29 September 1998 (1998-09-29) column 6, line 44 - line 61 column 28, line 43 - line 49 column 31, line 1 - column 34, line 22 ----- -/--	1-22

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search 27 September 2005	Date of mailing of the international search report 07/10/2005
Name and mailing address of the ISA European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Authorized officer Thibaudeau, J

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 02/20774

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	WO 00/45282 A (BOPS INCORPORATED) 3 August 2000 (2000-08-03) page 3, line 9 - line 28 page 8, line 7 - last line page 14, line 9 - page 17, last line	1-22
A	US 5 903 760 A (FARBER ET AL) 11 May 1999 (1999-05-11) column 2, line 4 - line 16	1-22

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No
PCT/US 02/20774

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 6026484 A	15-02-2000	NONE	
US 5815723 A	29-09-1998	NONE	
WO 0045282 A	03-08-2000	EP 1196855 A1	17-04-2002
		JP 2003520360 T	02-07-2003
US 5903760 A	11-05-1999	AU 4144697 A	14-01-1998
		WO 9750031 A1	31-12-1997